基于萤火虫优化的副本放置方法 *

李　君，侯孟书

(电子科技大学 计算机科学与工程学院，成都 611731)

摘　要：在云计算环境下分布式存储系统中，通常采用副本技术保证存储系统的可用性和可靠性，放置策略是副本技术的一个关键问题。针对现有副本放置策略中存在的副本访问开销大的问题，提出一种基于离散型萤火虫优化的副本放置算法。考虑副本放置对用户访问性能的影响，对其建立数学模型，计算萤火虫位置的适应度函数，并朝着荧光素值最大即最优值移动，进而得到合适的副本放置节点。通过仿真实验评估提出的方法，并与基于蚁群算法的副本放置策略进行比较。实验结果证明该算法能够选择合适的副本放置节点，具有较好的收敛性，并有效地降低存储系统的副本访问开销。

关键词：副本放置；萤火虫优化算法；云计算；分布式存储

中图分类号：TP393　　doi: 10.3969/j.issn.1001-3695.2017.09.0948

Replica placement strategy based on glowworm swarm optimization

Li Jun, Hou Mengshu

(*School of Computer Science & Engineering*, *University of Electronic Science & Technology of China*, *Chengdu* 611731, *China*)

Abstract: Cloud storage system often adopts replica technique to guarantee availability and reliability, and replica placement is a key issue. In order to solve the problem of high access overhead to replicas, this paper proposed a replica placement algorithm based on discrete glowworm swarm optimization. Through mathematical mode establishment for user access overhead, this algorithm computed the fitness function of glowworm position, updated individual position and then obtained appropriate nodes for replica placement. It conducted several simulations and compared with the replica placement strategy based on ant algorithm. The results show that the proposed algorithm can select appropriate nodes for replica placement, have a better convergence and reduce the replica access overhead.

Key Words: replica placement; glowworm swarm optimization; cloud computing; distributed storage

0　引言

随着互联网技术的发展，需要处理和存储的数据量呈爆炸性增长。在海量数据时代，传统的分布式存储解决方案具有成本过高、扩展性较差、难以提供持久有效的存储服务等缺陷。为了解决这些问题，云计算环境下的分布式存储技术应运而生。云计算环境下的分布式存储，通过集群应用，网格技术或分布式文件系统等功能，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作，共同对外提供数据存储和业务访问功能[1]。

云计算环境具有数据量大、存储设备异构以及网络状态差异等特点，数据丢失和出错成为云计算环境下分布式存储的一种常态。副本技术是解决此问题的一种有效方法，将数据的副本保存在多个不同的存储节点上，能够改善存储系统的可靠性和可用性。其中，放置策略是副本技术所涉及的关键问题，对用户的访问性能、存储系统的负载均衡、网络带宽的利用率以及副本之间的一致性维护有重要影响[2,3]。

针对云计算环境下分布式存储系统中副本放置问题，在学术界和工业界，研究人员进行了大量的研究工作，并取得了一系列的成果。文献[4]说明副本放置问题是一个 NP 难问题，基于图论，提出一种规约算法解决副本的放置，减小访问副本响应时间开销。文献[5]提出了一种分布式贪婪副本放置机制，根据数据对象的请求频度和系统容量放置副本，减少数据平均访问时间，并且从理论上证明该方法是 2 近似解决方案。对于故障和节点失效，文献[6]提出副本重新构建策略，解决存储数据副本的节点失效后，访问性能降低的问题。文献[7]研究采用基于超时值的副本一致性维护机制的副本放置问题，并给出此时求解最优放置策略的算法。文献[8]提出一种自适应的副本管理机制，通过对热度高的数据创建新副本并将其放置在负载较轻的节点上，减少访问响应时间。基于云的内容分发网络作为一

基金项目：国家自然科学基金面上项目（61472067）；国家科技支撑计划资助项目（2013BAH33F02）；四川省科技计划项目（2013GZ0006）

作者简介：李君（1988-），女，山东曹县人，博士研究生，主要研究方向为分布式存储、云计算（suiyuanlj2006@gmail.com）；侯孟书（1971-），男，教授，博士，主要研究方向为无线传感网络、分布式存储、移动计算.

个有效的内容分发体系架构提供内容分发服务，改善 QoS，扩展性和资源有效性。对于副本服务器放置问题，文献[9]提出贪婪启发式的方法，最小化操作开销和满足 QoS。文献[10]针对传统的服务-存储模式的集群 VOD 系统，提出一种自适应的媒体副本放置算法，该算法在不同数据量的情况下具有较好的负载均衡性和整体性能。为了解决数据网格环境下的副本放置问题，文献[11]提出将放置节点组织成树型结构，对负载均衡和副本数量进行研究。文献[12]基于社会网络特征，提出一种云环境下的服务副本放置策略SNPBRA，对部分云服务进行副本操作，减少异地服务间的交互，提高业务流程的执行效率。文献[13]针对教育网格中数据资源共享问题中的数据副本方面进行研究，设计一种合理的副本目录管理模型，并提出动态的副本创建策略，能够有效地提高副本定位的效率，适应用户请求的动态变化。除此之外，一些群体智能算法，比如粒子群算法，蚁群算法也用于解决存储系统副本的问题[14,15]。

萤火虫优化算法是一种基于群体智能的算法，其基本思想是通过模拟自然界中的萤火虫发光行为来搜索周围发光更亮的同伴，逐渐朝着区域内较优的位置移动，从而聚集到最亮的位置，实现搜索最优解的功能。萤火虫优化算法在寻找最优解过程中每个萤火虫相互独立，因此算法可以并发运行。此外，萤火虫优化算法避免了其他智能算法中早熟收敛的潜在缺点，萤火虫优化算法的收敛速度随着迭代次数而加快，使得其能够处理聚类和分类、组合优化等问题。萤火虫优化算法概念简单，易于实现和操作，具有较高的收敛速度和寻优精度，在很多领域得到了广泛的应用，具有良好的应用前景。

通过对萤火虫优化算法和副本放置问题进行分析，本文提出一种基于离散型萤火虫优化的副本放置算法。综合考虑副本放置影响因素，通过萤火虫个体的适应度值来计算其荧光素值，并且通过萤火虫个体间的移动概率，最终移动到荧光素最亮的萤火虫个体周围，进而找到最优解，选择合适的副本放置位置。

1　萤火虫优化算法

萤火虫优化算法(glowworm swarm optimization, GSO)是印度学者 Kaipa 等人[16]提出的一种新型群智能优化算法，它能够实现在求解空间寻优的目的。算法思想源于自然界中萤火虫晚上群聚活动这一自然现象，每个萤火虫通过自身所散发出的荧光素与其同伴进行寻觅食物，求偶等信息交流。萤火虫所携带的荧光素越强，它的号召力也就越强，最终会出现很多萤火虫个体聚集在荧光素较亮的萤火虫周围。

利用 GSO 求解问题时，首先将萤火虫个体随机地散布在求解空间中，每个个体都带有等量的荧光素值 l_0。算法经过多次迭代过程，每次迭代萤火虫个体 i 的荧光素值 l_i 都做相应更新，荧光素值取决于萤火虫所在位置的适应度函数值 $J(x_i(t))$。萤火虫个体与决策域半径内的萤火虫比较荧光素值大小，并依照概率机制向较亮的萤火虫个体移动。具体来说，GSO 算法按照下列步骤进行。

a) 初始化萤火虫 $i(i=1,2,\cdots n)$，将其随机放在目标函数的搜索范围内，并对各参数进行初始化。

b) 更新荧光素值。利用式(1)把萤火虫 i 在 t 次迭代时的位置 $x_i(t)$ 对应的适应度函数值转换为荧光素值 $l_i(t)$。其中：ρ 表示荧光素挥发因子，γ 表示荧光素增强因子，$\rho,\gamma\in[0,1]$。

$$l_i(t)=(1-\rho)l_i(t-1)+\gamma J(x_i(t)) \tag{1}$$

c) 确定邻域集合。在 t 次迭代时刻，第 i 只萤火虫的动态决策域半径内，选择荧光素值大于 i 的个体 j 组成其邻域集合 $N_i(t)$。其中：$d_{i,j}(t)$ 是萤火虫个体 i 与 j 之间的距离。

$$N_i(t)=\{j:d_{i,j}(t)<r_d^i(t);l_i(t)<l_j(t)\} \tag{2}$$

d) 计算移动概率。按照式(3)计算 t 次迭代时刻第 i 只萤火虫向其邻域中个体 j 移动的概率 $P_{ij}(t)$。

$$P_{ij}(t)=\frac{l_j(t)-l_i(t)}{\sum_{k\in N_i(t)}(l_k(t)-l_i(t))} \tag{3}$$

e) 更新位置。利用式(4)完成对移动后第 i 只萤火虫的位置更新。其中：s 是移动步长。

$$x_i(t+1)=x_i(t)+s*\left[\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}\right] \tag{4}$$

f)更新动态决策域半径。利用式(5)对动态决策域半径进行更新。其中：r_s 是萤火虫个体的最大感知半径；β 为动态决策域更新率；n_t 为个体邻域集内包含的萤火虫数目阈值；$|N_i(t)|$ 为萤火虫个体 i 邻域内萤火虫个体的数目。

$$r_d^i(t+1)=\min\{r_s,\max\{0,r_d^i(t)+\beta(n_t-|N_i(t)|)\}\} \tag{5}$$

2　基于萤火虫优化的副本放置算法

在云计算环境下分布式存储系统中，假设节点数为 N，所有节点通过分布式的方式连接，每个节点 $i\in N$ 最多保存一份数据的副本，存储系统中数据副本的数目为 R，则存储系统中副本放置问题可以描述为，选择合适的副本放置节点集合，能够处理数据副本的访问服务，同时保证副本访问的整体开销最小化。

根据以上对副本放置问题的定义，结合萤火虫优化算法的核心思想，本文提出一种基于萤火虫优化的副本放置方法。在对该问题建模阐述副本放置算法之前，先做如下假设：

a) 云计算环境下的分布式存储系统中，节点的拓扑结构固定。存储系统中的节点是异构的，每个节点自身的性能不同，具有一定的失效概率。

b) 为了简化算法，假设节点一次只能访问一个数据副本。在以后的研究工作中，将对本文算法进行扩展，支持节点并行访问多个数据副本。

c) 以节点之间的二维距离来度量访问数据副本所消耗的网络带宽。

2.1　副本放置模型

基于副本放置问题的定义和假设，对该问题进行建模，详细介绍该算法的数学描述及其参数含义。首先，定义该问题的目标函数，即副本访问的代价函数，如式(6)所示，其目标是保

证整体副本访问代价最小。因此，当该表达式取得最小值时，所选择的节点即为所求的副本放置节点。

$$\min C=\sum_{i\in N}\sum_{j\in D_i}\frac{1}{Perf_i}r_jDis_{ij}p_{ij} \tag{6}$$

其次，给出该目标函数的约束条件。

$$i,j\in[1,2,\cdots,N] \tag{7}$$

$$D_i=\{j\mid Dis_{ij}\le Distance,i\ne j\} \tag{8}$$

$$Dis_{ij}=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \tag{9}$$

$$r_j,p_{ij}\in[0,1] \tag{10}$$

$$\sum_{j\in D_i}r_j=R \tag{11}$$

$$\sum_{j\in D_i}p_{ij}=1 \tag{12}$$

其中：式(6) $Perf_i$ 表示节点自身的性能。式(7)定义了系统中的存储节点编号，节点总数为 N。式(8)表示给定一个存储节点 i，到该节点的距离小于 Distance 的节点集合。式(9) Dis_{ij} 表示任意两个节点 i 和节点 j 的距离。式(10)中 r_j 表示存储节点是否被选为副本放置节点，如果被选中置为 1，否则置为 0； p_{ij} 如果为 1，则节点 j 为节点 i 提供副本访问，如果为 0，则节点 j 不为节点 i 提供副本访问。式(11)表示所选择的副本放置节点数目为 R。式(12)表示为每个存储节点提供副本访问的节点只有一个。

2.2 基于离散萤火虫优化的副本放置算法

基于萤火虫优化算法原理，本节从解构造和编码、个体间距离度量、位置更新方法和适应度函数构造四个方面阐述基于萤火虫优化的副本放置算法。

2.2.1 解的构造和编码

基本萤火虫算法适用于连续型论域，将其进行离散化处理，对副本放置问题求解。本文采用整数编码方式表示一个可行解，比如编码为[2,6,9,18,26]，每一个整数为副本放置节点的编号。每一个可行解为 R 维序列，且序列中节点编号唯一。

2.2.2 个体间距离度量

由于副本放置问题是离散组合优化问题，萤火虫个体位置是以编码的方式构造的，其解向量中的变量为整数值。因此，不能采用基本萤火虫优化算法的欧氏距离计算个体之间的距离，本文通过编码的差异度来度量个体间距离[17]。对于 i、j 两个萤火虫个体，定义其第 t 次迭代时个体间的距离为

$$D_{i,j}(t)=C*\delta_{i,j}(t) \tag{13}$$

其中：C 为常数；$\delta_{i,j}(t)$ 为萤火虫个体 i、j 在第 t 次迭代时表示的编码序列的差异度，则差异度的计算方法如式(14)所示。

$$\delta_{i,j}(t)=\frac{\sum_{k=1}^{R}\left|d_{i,j}(t,k)\right|}{M} \tag{14}$$

其中：$\left|d_{i,j}(t,k)\right|$ 表示两个编码序列 $\boldsymbol{x}_i(t)$；$\boldsymbol{x}_j(t)$ 上同一位编码相差的绝对值； M 是 $\sum_{k=1}^{R}\left|d_{i,j}(t,k)\right|$ 取值的上限，当 N 为偶数时，$M=\frac{NR}{2}$，当 N 为奇数时，$M=\frac{(N^2-1)R}{2N}$。显然，$\delta_{i,j}\in[0,1]$ 且 $\delta_{i,j}=\delta_{j,i}$，本文 C 取值为 20。

2.2.3 位置更新方法

由于副本放置的离散性，式(4)不再适用个体位置更新。假设分布式存储系统中节点数目为 N，需要选择 R 个作为副本放置节点。副本放置问题的解结构为 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{iR})$，$i=1,2,\cdots,n$，其中 n 为萤火虫个体数目。即每个萤火虫对应一个 R 维行向量，行向量中的每一维按照不同概率选择相应的位置更新公式，从而实现对解向量的更新。

$$x_{ik}(t+1)\begin{cases}x_{ik}(t), & \text{if } r(k)<p_1\\ x_{jk}(t), & \text{if } p_1\le r(k)<p_2\\ Int((N-1)*rand)+1, & \text{if } r(k)\ge p_2\end{cases} \tag{15}$$

其中：$\boldsymbol{r}$ 是从 0 到 1 之间随机产生的 R 维序列；$x_{jk}(t)$ 是萤火虫个体 i 在第 t 次迭代时选择移动对象的位置； p_1 和 p_2 为更新选择概率参数，p_1，$p_2\in[0,1]$ 且 $p_1<p_2$。由式(15)可知，萤火虫个体 i 以概率 p_1 接受 $\boldsymbol{x}_i(t)$ 编码的第 k 维数据 $x_{ik}(t)$，以概率 p_2-p_1 接受 $\boldsymbol{x}_j(t)$ 编码中的第 k 维数据 $x_{jk}(t)$ 以概率 $1-p_2$ 随机更新其第 k 维变量。$Int((N-1)*rand)+1$ 是保证随机产生的变量在 $[1,N]$ 内。

在萤火虫优化算法迭代过程中，某些解向量中的变量可能不是唯一的。为了保持种群的规模和多样性，在一轮迭代后，检查解向量中唯一变量的数量 m，如果小于 R，则随机产生 $R-m$ 个唯一变量 $x_{ik}\in[1,N]$，将可行解代替非可行解。这样使得每次迭代时都能保证搜索空间中的解为可行解，由于随机性而导致的收敛到次优解的概率也会降低。

为了解决局部最优的问题，在算法后期随机产生一个整数 i($0<i<R$)，对于种群中目标函数值较好的个体和较差的个体，随机产生长为 i 的两个片段进行交换。如果新个体的目标函数值比原个体优，则新个体替换原个体，否则保留原个体。

2.2.4 适应度函数

适应度函数 $J(x_i(t))$ 是根据萤火虫个体 i 在第 t 次迭代的编码 $\boldsymbol{x}_i(t)$，按照式(16)得到的值求其倒数。其中 F 取一个特别大的正数作为惩罚系数，适应度函数考虑了节点的失效概率，如果在一段时间内可能失效的概率大于系统中预先设定的节点失效概率阈值 Tolerance，那么适应度函数的值将减小，即该节点被选择作为副本放置节点的可能性将大大降低。节点失效概率阈值可以根据存储系统的应用需求而设定。

$$\sum_{i\in N}\sum_{j\in D_i}\frac{1}{Perf_i}r_jDis_{ij}p_{ij}+F\sum_{i\in N}\sum_{j\in D_i}\max((Failure_j-Tolerance),0) \tag{16}$$

2.3 基于离散型萤火虫优化的副本放置算法步骤

本文提出的基于离散型萤火虫优化算法解决副本放置问题的具体步骤如下所示：

a)对算法基本参数进行设置。

b) 初始化萤火虫种群。设置分布式存储系统中节点总数 N，副本放置节点的数目 R 即每只萤火虫的维数为 R，每一维取值

为$[0, N]$区间内的随机整数，并将记录迭代次数的变量 $iter$ 置为 0。

c)根据适应度函数公式，计算每只萤火虫个体对应解的适应度，进一步获得相应的荧光素值。

d)计算萤火虫个体之间的距离，每只萤火虫个体在其动态决策半径内，选择荧光素值比自己大的萤火虫个体组成其邻域。

e)通过移动概率公式计算萤火虫个体 i 向个体 j 移动的概率，且 i 朝概率 $P_{ij}(t)$ 最大的个体 j 移动。

f) 随机产生一个在$[0,1]$区间的 R 维向量 $\boldsymbol{r}$，根据向量 $\boldsymbol{r}$ 每一维的值，按照式(15)更新解向量中的每一维变量，并对非可行解做相应处理。

g) 根据式(5)对萤火虫个体的决策域半径进行更新。

h) 判断是否满足 $iter > Iter_max$，若满足条件则算法结束，输出全局最优值及其对应的位置；否则转到 c)。

2.4　时间复杂度分析

设萤火虫数目为 n，副本放置节点的数目为 R，迭代次数为 $Iter_max$。初始化解的时间复杂度为 $O(n)$，计算 n 个个体荧光素时间复杂度为 $O(n)$，计算个体间距离的时间复杂度为 $O(n^2R)$，一次迭代中，萤火虫位置更新时间复杂度为 $O(n^2+nR)$。所以算法在迭代次数之内的时间复杂度为 $O(Iter_max(n^2R))$。

3　实验评估

为了验证本文提出的基于离散型萤火虫优化的副本放置算法，首先利用 MATLAB 验证算法的有效性和收敛性。对 CloudSim 进行扩展，通过修改 Host 和 Datacenter 类，添加新的副本放置策略组件完成副本放置工作，比较分析算法的副本访问开销。实验环境为：软件采用 Windows 7 操作系统，MATLAB R2014a，CloudSim3.0.3，硬件采用 i5-3470 @3.2 GHz，内存为 8 GB。本实验中算法涉及的参数如表 1 所示。

表 1　实验参数的取值

参数	取值
迭代次数	200
种群规模	80
荧光素挥发因子和增强因子	ρ=0.4，γ=0.6
动态决策域更新率	0.08
荧光素初始值	5
动态决策域半径初始值	6
个体最大感知半径	20
邻域集阈值	5
位置更新公式选择参数	$p_1=0.5, p_2=0.8$

本实验采用了国内省会城市及直辖市的地理坐标并作规范化处理，代表存储系统中不同节点的相应位置。在[0.01,0.05]内为每个节点随机分配一个失效概率，根据经验取值设定系统中节点失效概率阈值 Tolerance 为 0.03。副本数量根据三副本法则设置，也可以根据实际分布式存储系统的应用需求进行调整。根据 Krishnanand 和 Ghose 所阐述的挥发因子和增强因子的取值，本文定义 ρ=0.4，γ=0.6。结合对萤火虫个体间距离度量的分析和萤火虫种群规模的选取，本文选择最大感知半径为 20。

3.1　副本放置节点的选择

图 1 是分布式存储系统中节点的分布情况以及采用本文提出的副本放置方法选取的副本放置节点位置。本文提出的方法根据节点自身的性能和节点失效概率，构建目标函数，选取目标函数最小时的节点作为副本放置位置。在副本放置位置的选择过程中，以副本整体访问开销最小化为目标，一旦确定副本放置节点，其整体的开销是最小的。这样便能够选择合适的节点放置副本，减小副本访问开销，提高用户的访问性能并均衡系统负载。

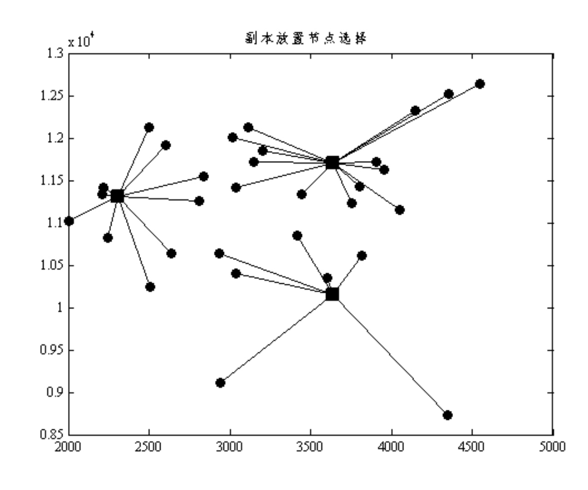


图 1　基于萤火虫优化的副本放置节点选择情况

3.2　算法收敛性

图 2 描述了基于萤火虫优化的副本放置算法的收敛曲线。从图中可以看出，算法较快地收敛到最优值，能够选择合适的节点进行副本放置，用户对副本访问的开销能够达到最小化。本文提出的基于萤火虫优化的副本放置算法能够高效的选择合适的副本放置节点，具有较好的收敛性。

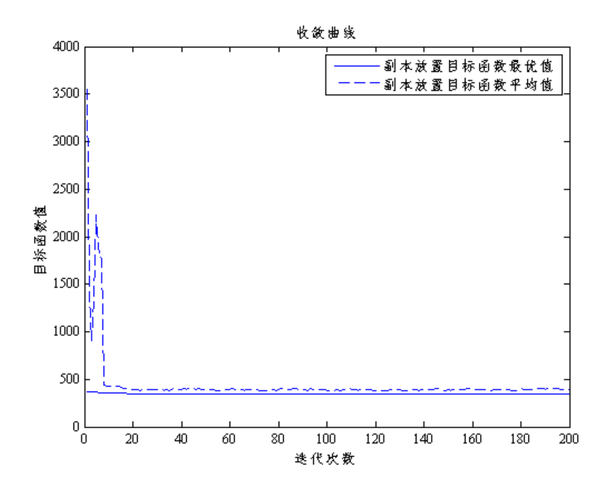


图 2　基于萤火虫优化的副本放置算法收敛曲线

3.3 参数的选择

图 3 描述了种群数量和迭代次数不同时算法的收敛曲线。可以看到，种群规模 n 影响算法目标函数值，当 n 大于 80 时具有较好的收敛性。图 4 和 5 分别描述了萤火虫种群数目和最大迭代次数对算法目标函数最优值的影响。分别在一定的种群规模和迭代次数情况下，进行 50 次实验，统计取得目标函数最优值的比例。从图 4 可以看出，开始增大萤火虫种群规模，能够提高算法的性能，随着种群规模继续增大，目标函数最优值比例会趋于稳定。

图 5 表示当种群规模取默认值时，迭代次数不会对算法性能产生较大的影响，随着迭代次数的增加，并不能明显提高目标函数最优值的比例。种群规模和迭代次数是算法时间复杂度的两个关键影响因素，应当选取合适的种群规模和迭代次数，适当地减少迭代次数，增加种群规模，在保证较高的算法性能的基础上，降低时间复杂度。

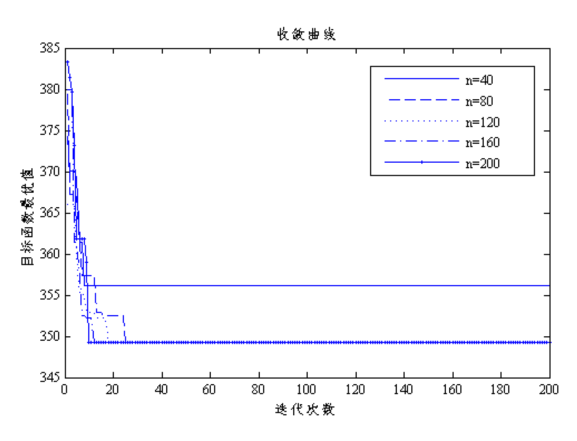


图 3 不同萤火虫种群数量目标函数收敛曲线

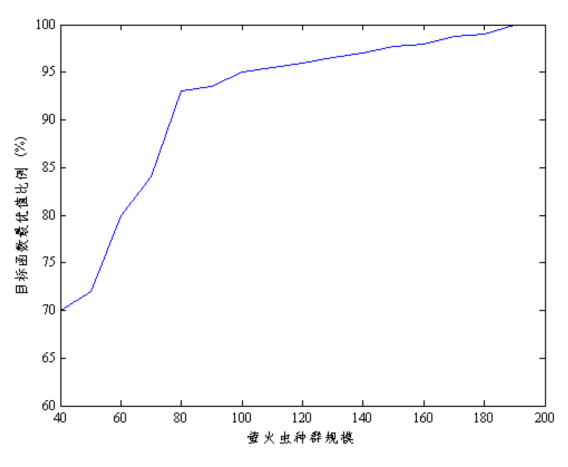


图 4 萤火虫种群规模对算法性能的影响

3.4 访问时间

图 6 描述了在 CloudSim 上对基于萤火虫优化与基于蚁群算法的副本放置策略[18]进行仿真实验，比较分析在不同的用户访问模式下的平均访问时间度量访问开销。从图中可以看出，顺序分布、均匀分布、高斯分布和齐普夫分布四种访问模式下，本文提出的基于萤火虫优化的副本放置算法的平均访问时间都要优于基于蚁群算法的副本放置策略。由于本文提出的方法在副本放置位置的选择过程中，以副本整体访问开销最小化为目标，基于萤火虫优化的副本放置算法能够选择较合适的节点进行副本放置，有效地减少副本访问时间，具有较好的性能。

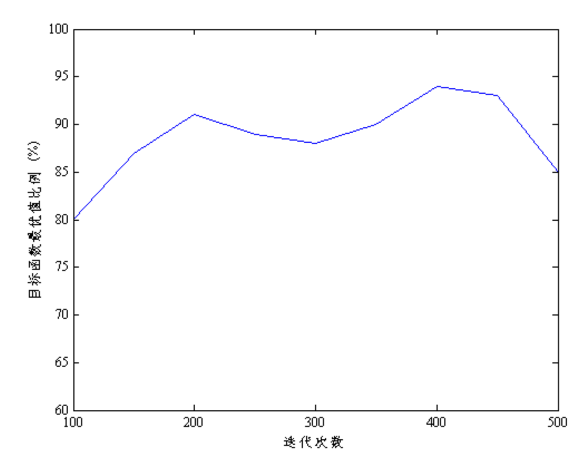


图 5 迭代次数对算法性能的影响

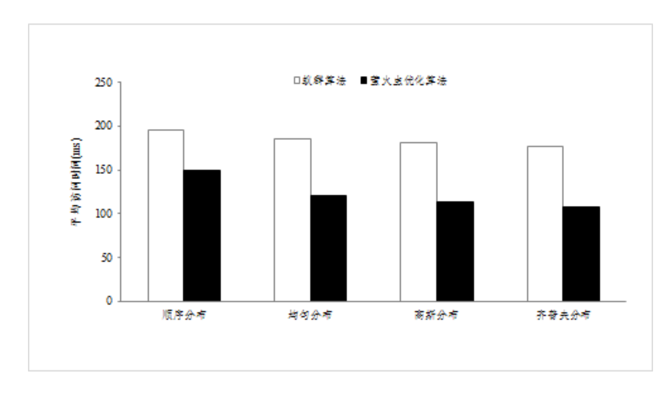


图 6 两种算法在不同访问模式下的平均访问时间

4 结束语

针对云计算环境下分布式存储的副本放置问题，本文提出一种基于离散型萤火虫优化的副本放置算法，选择副本放置的最佳位置。全面考虑副本放置节点对用户访问性能的影响，建立数学模型，设计合理的编码方式，采用编码差异度来度量萤火虫个体之间的距离，并引入萤火虫位置更新和不可行解的处理方法。通过仿真实验验证了本文提出方法的有效性，实验结果表明该方法降低了用户访问副本的整体开销，具有较好的收敛性。在接下来的工作中，将在实际分布式存储环境中实现该方法，进行更好的实验验证和副本放置策略研究。

参考文献：

[1] Mell P M, Grance T. The NIST definition of cloud computing. National Institute of Standards and Technology [R]. Gaithersburg: National Institute of Standards and Technology, 2011.

[2] Sun Dawei, Chang Guiran, Gao Shang, et al. Modeling a dynamic data replication strategy to increase system availability in cloud computing

environments [J]. Journal of Computer Science and Technology, 2012, 27 (2): 256–272.

[3] Ye Zhen, Hu Weijian. An adaptive replica selection algorithm for quorum based distributed storage system [J]. International Journal of Grid and Distributed Computing, 2016, 9 (5): 55-68.

[4] Bai Xiaohu, Jin Hai, Liao Xiaofei, et al. RTRM: a response time-based replica management strategy for cloud storage system [C]// Proc of International Conference on Grid and Pervasive Computing. Heidelberg: Springer, 2013: 124-133.

[5] Zaman S, Grosu D. A distributed algorithm for the replica placement problem [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22 (9): 1455-1468.

[6] Higai A, Takefusa A, Nakada H, et al. A study of effective replica reconstruction schemes for the hadoop distributed file system [J]. IEICE Trans on Information and Systems, 2015, E98. D (4): 872-882.

[7] Tang Xueyan, Chanson S T. Analysis of replica placement under expiration-based consistency management [J]. IEEE Trans on Parallel and Distributed Systems, 2006, 17 (11): 1253-1263.

[8] Chen Lingfeng, Hoang D B. Adaptive data replicas management based on active data-centric framework in cloud environment [C]// Proc of IEEE International Conference on High Performance Computing and Communications and IEEE International Conference on Embedded and Ubiquitous Computing. New York: IEEE Press, 2013: 101-108.

[9] Sahoo J, Glitho R. Greedy heuristic for replica server placement in cloud based content delivery networks [C]// Proc of IEEE Symposium on Computers and Communication. New York: IEEE Press, 2016: 302-309.

[10] 赵俊，金海. 自适应的集群流媒体文件副本放置策略 [J]. 计算机应用研究, 2008, 25 (2): 594-596.

[11] Wu Janjan, Lin Yingfang, Liu Pangfeng. Optimal replica placement in hierarchical data grids with locality assurance [J]. Journal of Parallel and Distributed Computing, 2008, 68 (12): 1517-1538.

[12] 罗浩宇，陈旺虎. 基于社会网络特征的云服务副本放置策略 [J]. 计算机应用, 2013, 33 (8): 2143-2146.

[13] 高田，刘方爱. 教育资源网格中的一种动态数据复制技术 [J]. 计算机应用研究, 2008, 25 (3): 869-871.

[14] Wang Lijuan, Luo Junzhou, Shen Jun, et al. Cost and time aware ant colony algorithm for data replica in alpha magnetic spectrometer experiment [C]// Proc of IEEE International Congress on Big Data. New York: IEEE Press, 2013: 247-254.

[15] Lim S P, Haron H. Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions [C]// Proc of IEEE Conference on Open Systems (ICOS) . New York: IEEE Press, 2013: 41-46.

[16] Kaipa K N, Ghose D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics [C]// Proc of IEEE Swarm Intelligence Symposium. New York: IEEE Press, 2005: 84-91.

[17] Singh A, Thapar S, Bhatia A, et al. Disk scheduling using a customized discrete firefly algorithm [J]. Cogent Engineering, 2015, 2 (1): 1011929.

[18] 沈薇，刘方爱. 基于蚁群算法的数据副本放置策略 [J]. 计算机应用研究, 2007, 24 (6): 82-84.